Final Exam, Spring 2013 **Monday May 16th, 2013** **Exam time: 01:00-4:00 P.M.**

Student's name:..... **ID:** ..... **Section:** .....

Problem 1 (7 points)

For each of the question below, circle either T (for **True**) or F (for **False**). **No explanations** are needed. Incorrect answers or unanswered questions are worth zero points.

T F Let T be the minimum spanning tree of a connected, undirected, and weighted graph G = (V, E). If e is a new edge, then $T \cup \{e\}$ contains a cycle.

T F Bellman-Ford algorithm works on all graphs with negative-cost edges.

T F Dijkstra's algorithm works on graphs with negative-cost edges.

T F Let G be an edge-weighted directed graph with source vertex s, and let T be a shortest path tree from s. If we add a positive constant p to the cost of every edge incident on s in G. T remains a shortest path tree from s.

T F If someone was able to give an exponential time lower bound for a problem that is *NP-complete*. Then this would imply that *P* is not equal to *NP*.

T F There exist a polynomial time algorithm to determine whether an undirected graph contains a clique of size 3.

T F Suppose problem P_1 can be reduced to problem P_2 in linear time (i.e.,$P_1 \propto O(n)$ P_2). Then, if there exists a polynomial time algorithm for P_1, there exists a polynomial time algorithm for P_2.

Problem 2 (9 points)

Solve the following recurrences using the Master theorem by giving tight θ-notation bounds. Justify your answers.

(a) $T(n) = 3T(n/5) + \log^2 n$

(b) $T(n) = 2T(n/3) + n \log n$

(c) $T(n) = 7T(n/2) + n^3$

Problem 3 (10 points)

Let $A_1, \ldots, A_4$ be matrices with dimensions $5 \times 2, 2 \times 4, 4 \times 5, 5 \times 10$, respectively. In finding an optimal parenthesization of the matrix chain product $A_1 * A_2 * A_3 * A_4$, we use two tables **m[,]** and **s[,]** below. Here **m[i,j]** stores the optimal cost of computing subchain $A_i..A_j$ and **s[i,j]** records the index k where the optimal parenthesization splits $A_i..A_j$ between A_k and A_{k+1} for some k with $\mathbf{i} \leq \mathbf{k} \leq \mathbf{j-1}$.

a- What is the recursive equation of **m[i,j]**?

b- Fill the empty entries in the two tables. Show your work in each case.

m

j \ i	1	2	3	4
4				0
3			0	
2		0		
1	0			

s

j \ i	1	2	3
4			
3			0
2		0	

c- Now, give the optimal parenthesization of the matrix chain product $A_1 * A_2 * A_3 * A_4$. Show how you came up with the solution using the tables above.

Problem 4 (9 points)

Solve the following instance of the Knapsack Problem using Dynamic programming paradigm. The maximum allowed weight is **W**max = **10**.

i	**1**	**2**	**3**	**4**
Vi	20	10	15	31
Wi	6	1	2	5

a- Give the recursive equation you used to define the data structure needed for your dynamic programming solution. Then, fill the proposed data structure.

b- What is your solution to this instance of the Knapsack problem?

Problem 5 (10 points)

A tow truck is moving through a straight one-way highway from location **A** to location **B**. The driver has received many requests from customers who need to tow their cars along this highway. Each customer **C** is defined by two numbers **C.location** and **C.destination**. The customer **C** is located along the highway at position **C.location** units away from **A**, and wants to tow his car along the highway to the location that is **C.destination** units away from **A** (**C.location** < **C.destination**). Customers are charged a fixed amount (100 SR) regardless of the distance. The tow truck cannot carry more than one car at the same time. Also, the driver cannot drive back.

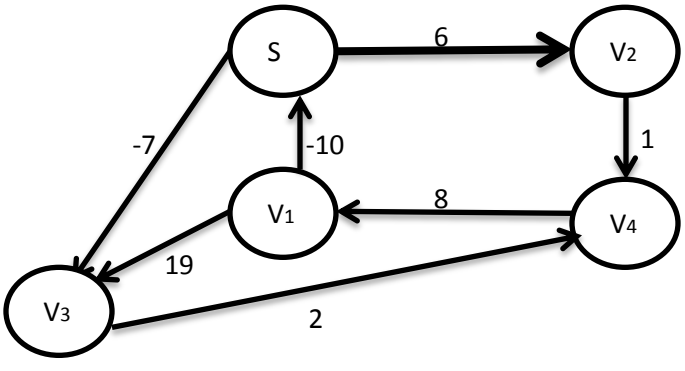
Assume that the customer information are available in an array **C**, i.e., **C[i].location** and **C[i].destination** are the pickup and the destination of the i^{th} customer, respectively.

a- Describe an algorithm that assists the tow truck driver to maximize his profit. The algorithm should print the **.location** and **.destination** properties of customer cars that should be towed.

b- What is the time complexity of your algorithm? Which programming design technique did you use?

Problem 6 (5 points)

Apply Bellman-Ford algorithm on the graph below to solve the single source shortest path starting from vertex S. You need to show the results of each step with all details (hint: use a table as we did in class). Also, you need to show the final results.



Problem 7 (10 points)

Given an undirected graph G = (V, E), a set of vertices W ⊆ V is a clique if for all u, v ∈ W, (u, v) ∈ E. In other words, there is an edge between every pair of vertices in W.

Given an undirected graph G = (V, E), a set of vertices W ⊆ V is an independent set if a for all u, v ∈ W, (u, v) ∉ E. In other words, there is NO an edge between any pair of vertices in W.

Consider the CLIQUE and INDEPENDENT SET (IS) problems for undirected graphs that we studied in class.

CLIQUE = {(G, k) | G has a clique of size k}

IS = {(G, k) | G has an independent set of size k}

We define next the new problem ISLIQUE

ISCLIQUE = {(G, k) | G has a clique of size k, and an independent set of size k}.

a- Define a certificate for ISCLIQUE. Show that we can verify the certificate in deterministic polynomial time.

b- Consider an undirected graph G and an integer k. Construct a new graph H from G by adding k vertices to G but no additional edges. So, G = (V, E) and H= (V U W, E) where W is a set of k new vertices.
   i. Show that if (G, k) ∈ CLIQUE then (H, k) ∈ ISCLIQUE.
   ii. Show that if (H, k) ∈ ISCLIQUE then (G, k) ∈ CLIQUE.

c- What have we shown in (a) and (b)? Using the fact that CLIQUE is NP-complete, what can we now conclude?